Introduction to Software Cost Estimation

Martha Ann Spurlock
BMC
14 May 2003

Objectives

- Become familiar with the state of software cost estimating in the DoD
- Identify the major problems with software cost estimating
- Discuss techniques for determining the size of the software
- List other factors and the influence they have on software cost estimating

Topics

- Software Problems
- Uncertainty in Software Estimates
- Software Growth in Weapon Systems
- GAO Reports
- Singing the Software Blues
- Sizing Techniques

More Influences

- Testing Considerations
- Reuse Considerations
- Capability Assessments
- Architecture/Interoperability Considerations
- COTS
- Language
- Maintenance
- Reality

Estimating Costs

- So, how are we doing?

DOD Software Costs

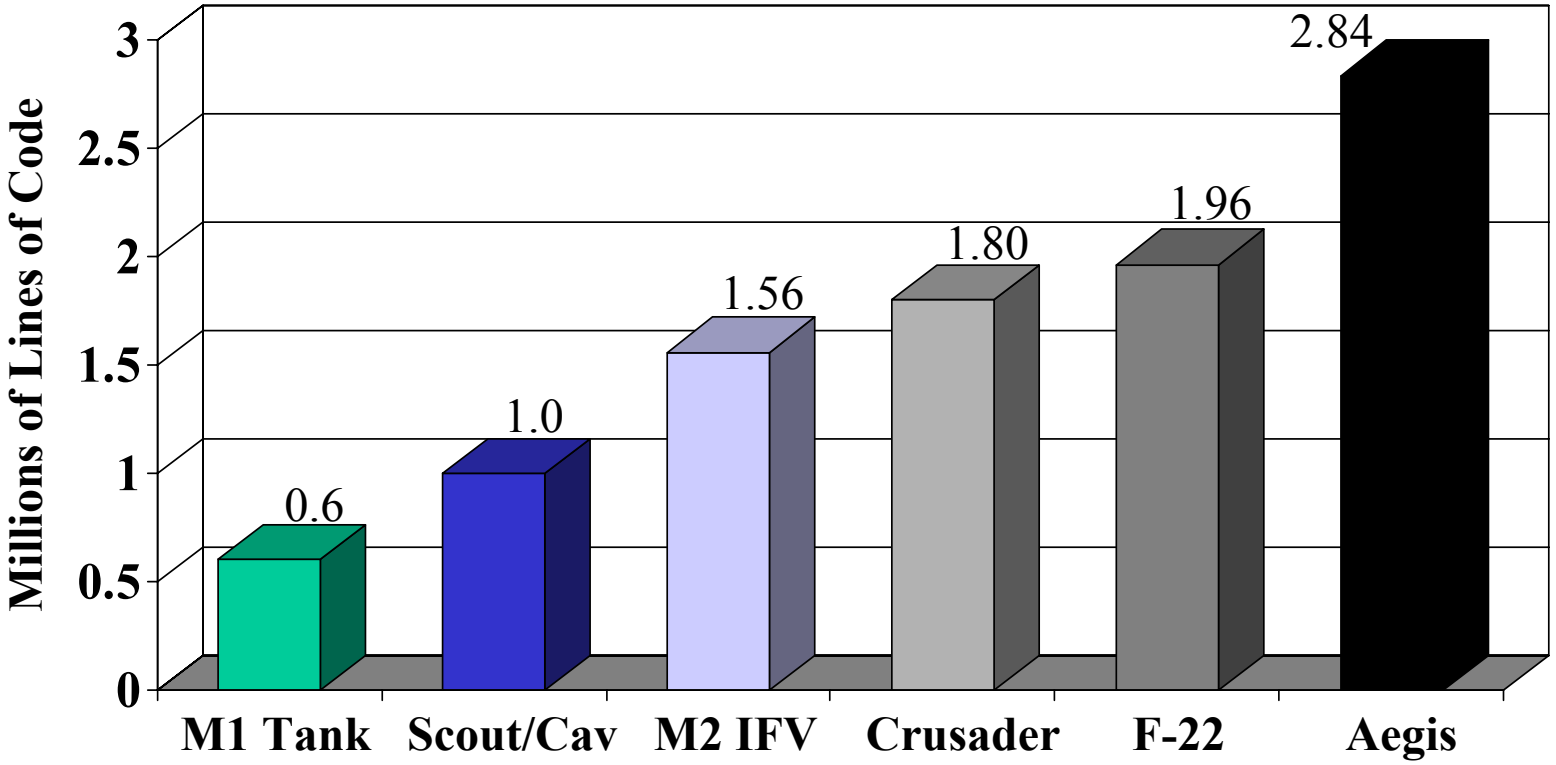
- Over the past 10 years, the defense budget has declined by 35% according to the GAO.
- In FY 92 DOD spent $35.2 billion on software intensive systems, $29.1 billion (83%) of which was for software alone.
- According to LTG Carl G. O'Berry, former Air Force Deputy Chief of Staff tells us that the FY95 budget reached $42 billion. This represents a 31% increase in just three years.

Software Problems

"Software problems have caused major delays of weapons systems, created malfunctioning aircraft, and cost the Department of Defense billions of dollars in unanticipated costs. Officials acknowledge that virtually every troubled system, from the electronics of the B-1B bomber to satellite tracking systems, has been affected with software problems. Even straightforward record-keeping systems can get bogged down; last year the Navy canceled a software accounting project nine years in the making after its cost quadrupled to $230 million."

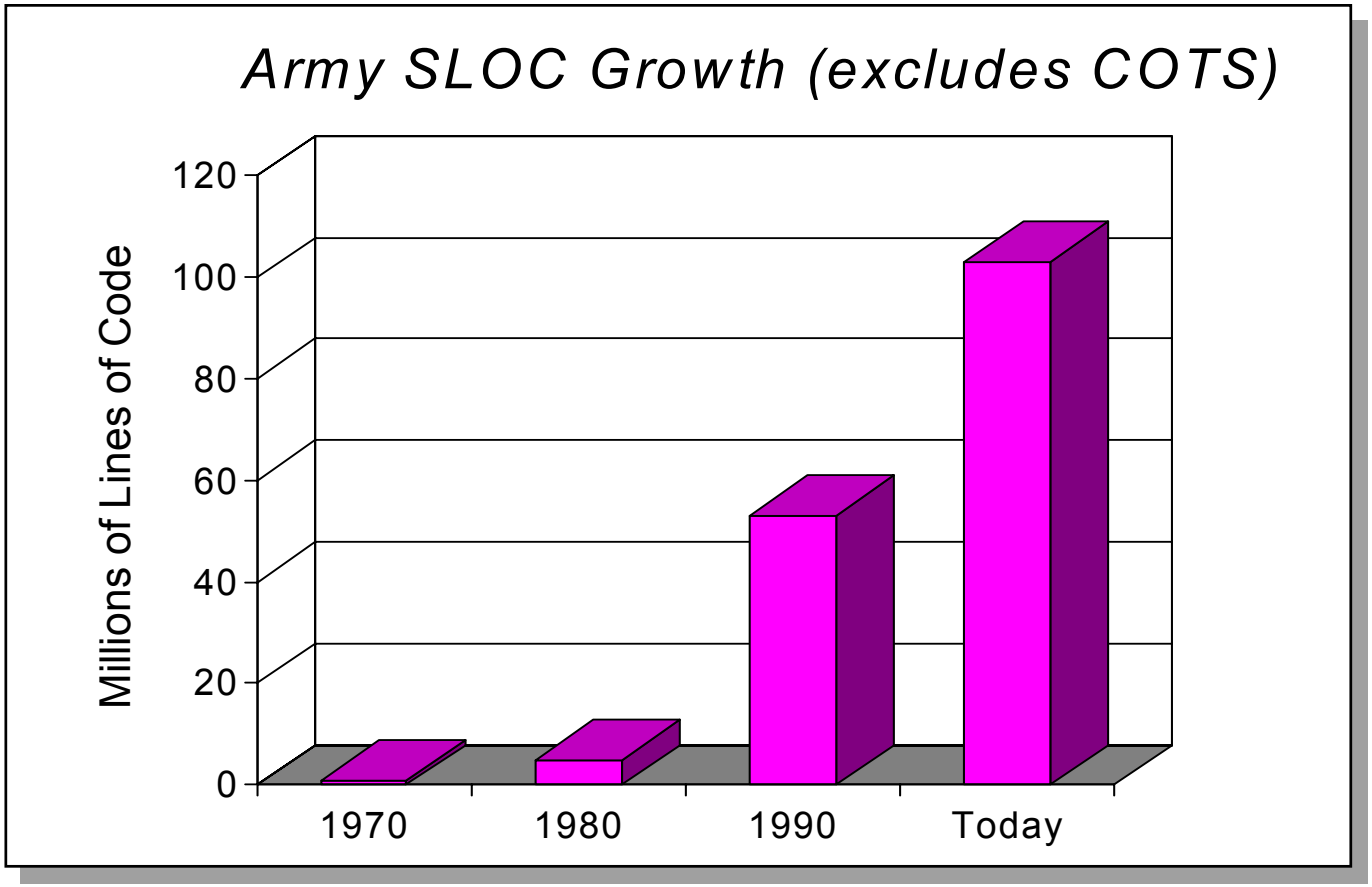
Evelyn Richards, "Society's Demands Push Software to Upper Limits:More Computer Crises Likely," The Washington Post, Dec 9,1990

Weapon System Software Size



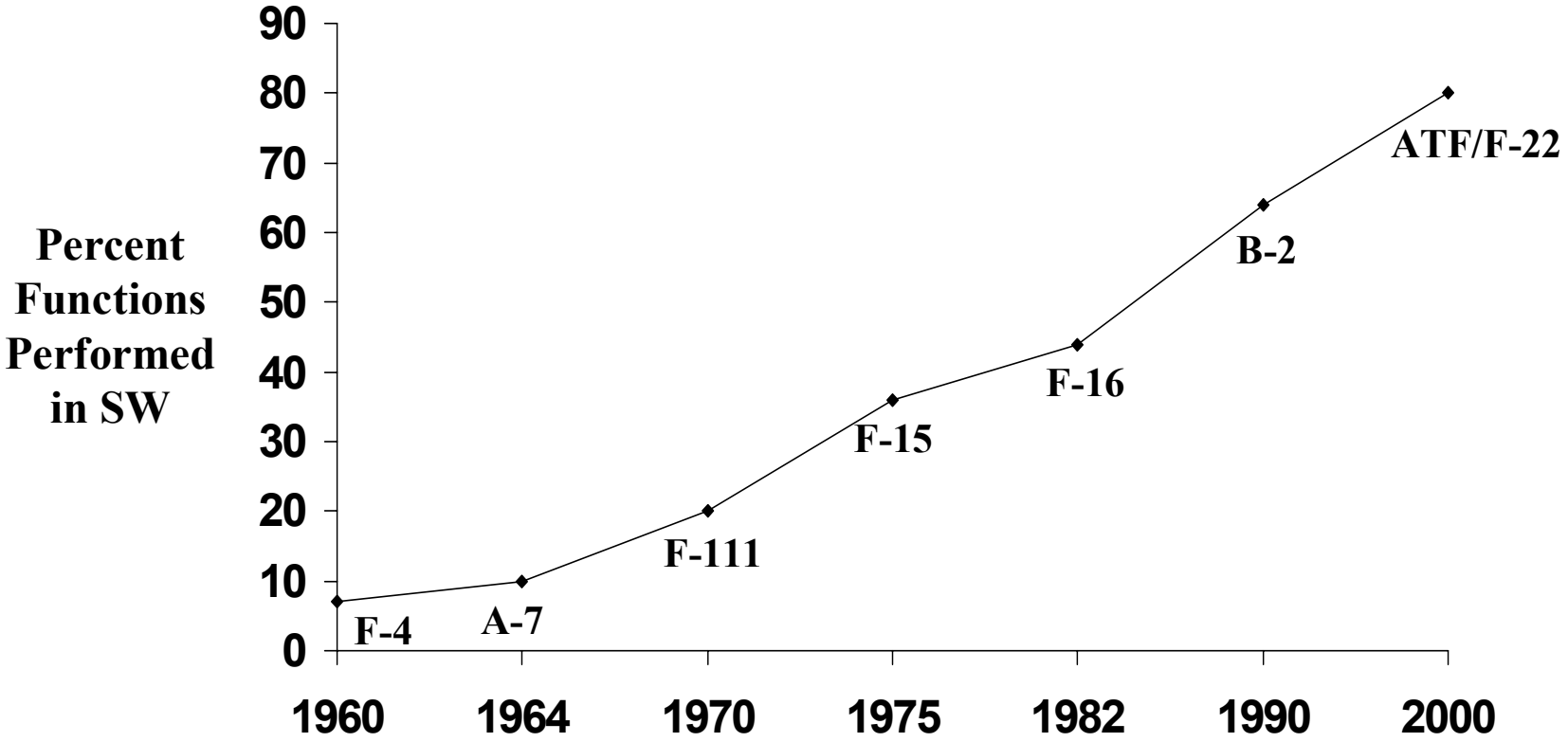
Source: Dr. Delores Etter's, DUSD(S&T), STC 1999 keynote address.

Software Growth



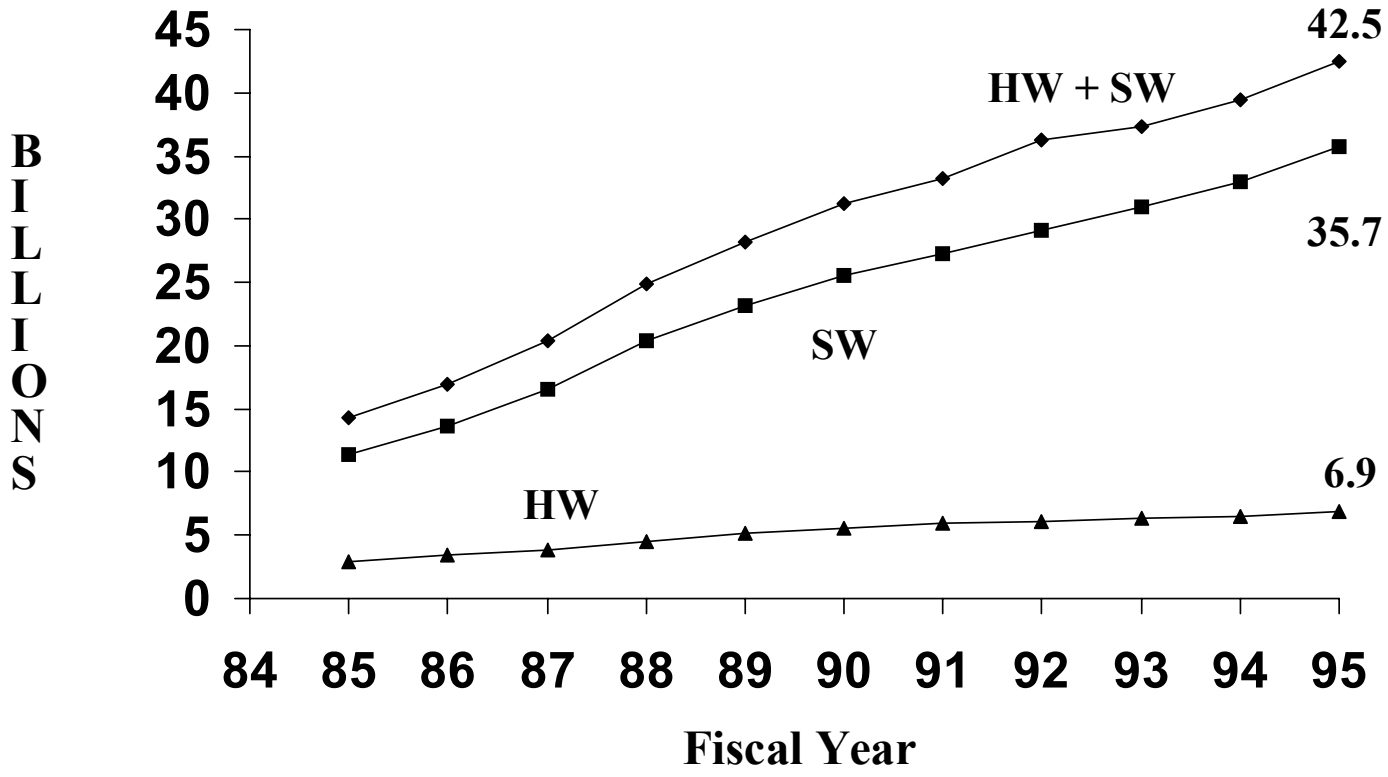
Source: DSMC ISAC Systems Engineering Student Guide, Oct-Dec 1998.

Weapon System Software Dependencies



Source: U.S. Air Force, "Bold Stroke" Executive Software Course, 1992.

DoD Information Systems Market



Currently spending approx. $50 billion a year on SW & SW intensive systems.

EIA study reported in volume 1 of the Sep 94 Moseman/Susan Johnson blue book.

GAO Reports

- Analysis of custom built MIS systems
- Sergeant York
- Navy ‘s Financial System
- C-17
- Software Challenges in Mission Critical DoD Systems

MIS Systems

163 contractors and 113 government personnel surveyed

- +60% of contracts had schedule overruns
- +50% had cost overruns
- +45% of software could not be used
- +29% of software was never delivered

Sergeant York

- 64 of planned 614 units delivered and subsequently scrapped
- Cost and schedule overruns
- $1.8 Billion lost
- Program canceled

Navy's Standard Automated Financial System

- $446.5 million (99.9%) projected cost overrun
- 5 year projected schedule overrun
- $230 million lost
- Program canceled

C-17

- $1.5 billion cost overrun
- Software size/complexity underestimated
- MilStds waived for contractor with limited software experience
- Shortcuts taken on software testing and software supportability issues

Mission Critical DoD Systems
15 Major Systems Studied

- Poor software engineering concepts
- Poor testing
- Security requirements not met
- Requirements ill defined

Air Traffic Control: Advanced Automation System

- 5 years behind schedule
- $2.6 billion cost overrun
- $238 million spent due to delays

Cheyenne Mountain Upgrade Program

- 8 years behind schedule (at the time of the report (ATR))
- $792 million over budget (ATR)
- 11 years projected schedule slip
- $896 million projected budget overrun
- $22 million/year for continued maintenance of old system

Scientific American Oct 1994

- IBM study of 24 companies developing large, distributed software intensive systems.
  - 55% had cost overruns
  - 68% had schedule overruns
  - 88% had to be redesigned to be used

Inadequate Estimates

The fundamental reason software-intensive developments overrun cost and schedule, resulting in quality and performance shortfalls, is our ***inability to estimate...*** We often forget that software development involves much more than simply writing code. For example, we are still learning that software inspections and testing take longer than anticipated and that maintenance consumes from **60%** to **80%** of our software dollars. We also do not account for the amount of scrap and rework of code involved... Boehm claims this cost to be about **44%** of each dollar spent...

The Software Blues

Dr. Patricia Sanders' (Director, Test, Systems Engineering and Evaluation, OUSD(A&T)) STC 1998 Keynote Address:

- **Only 16% of SW development will finish on time and on budget.**
- **Rework = 40% of SW development costs.**

  $45B x 40% = *$18B annual loss*

⟹ **Best in Class – Rework = 8 to 11 %**

$45B x 8% = $3.60B - $18B = $14.40B annual savings

$45B x 11% = $4.95B - $18B = $13.05B annual savings

Cause 1

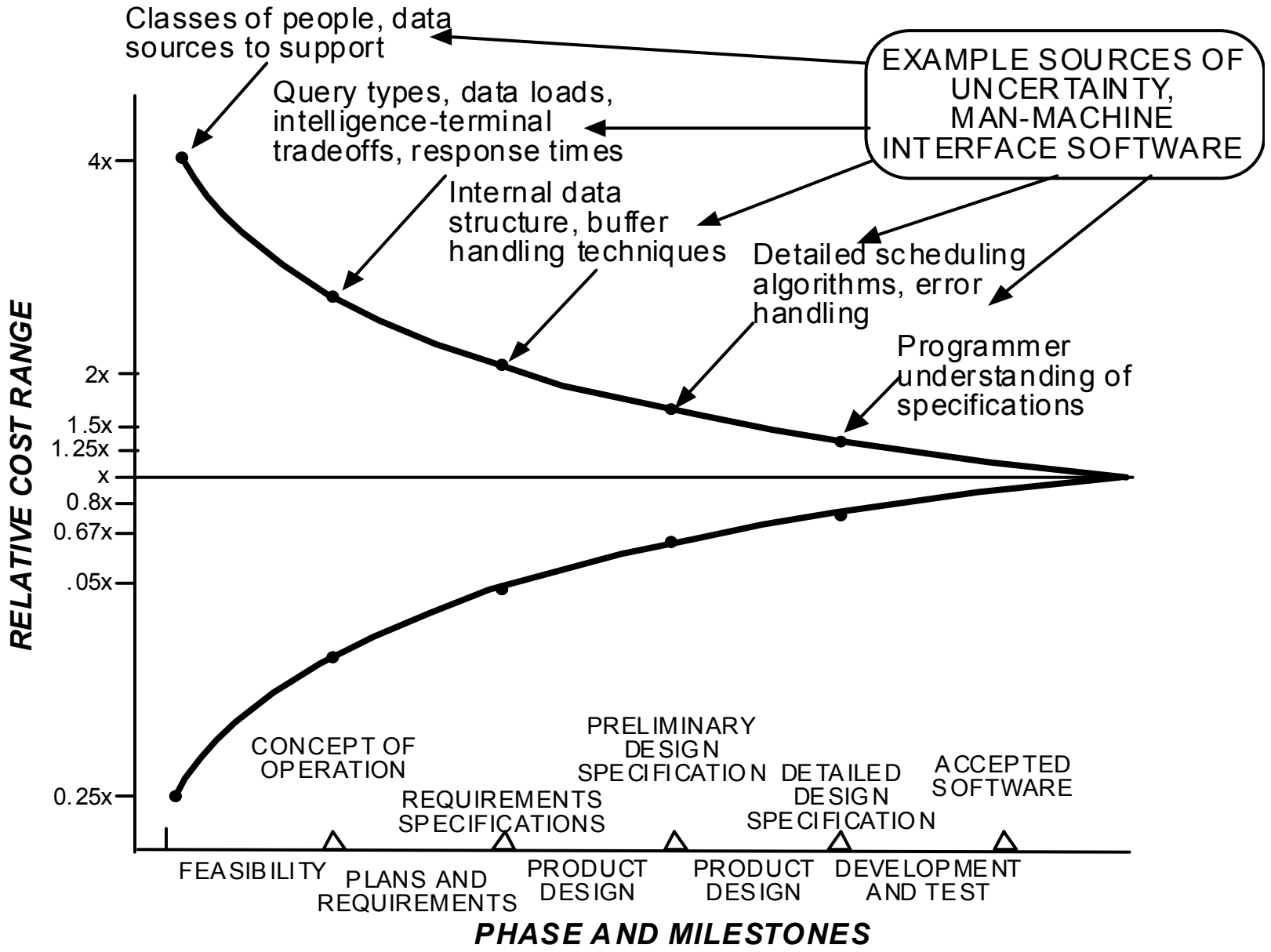
- **Inadequate Requirements Determination:**
  - **Users don't know what information they need:**
    - **GAO Study:** **+45%** of SW couldn't be used and **+29%** of SW was never delivered.
    - **IBM Study:** **88%** had to be redesigned to be used.
    - **Standish Group Study:** Challenged projects delivered with only **61%** of originally specified functions.
  - **Rome Lab indicates that over 50% of all SW errors are requirements errors.**

Requirements Instability

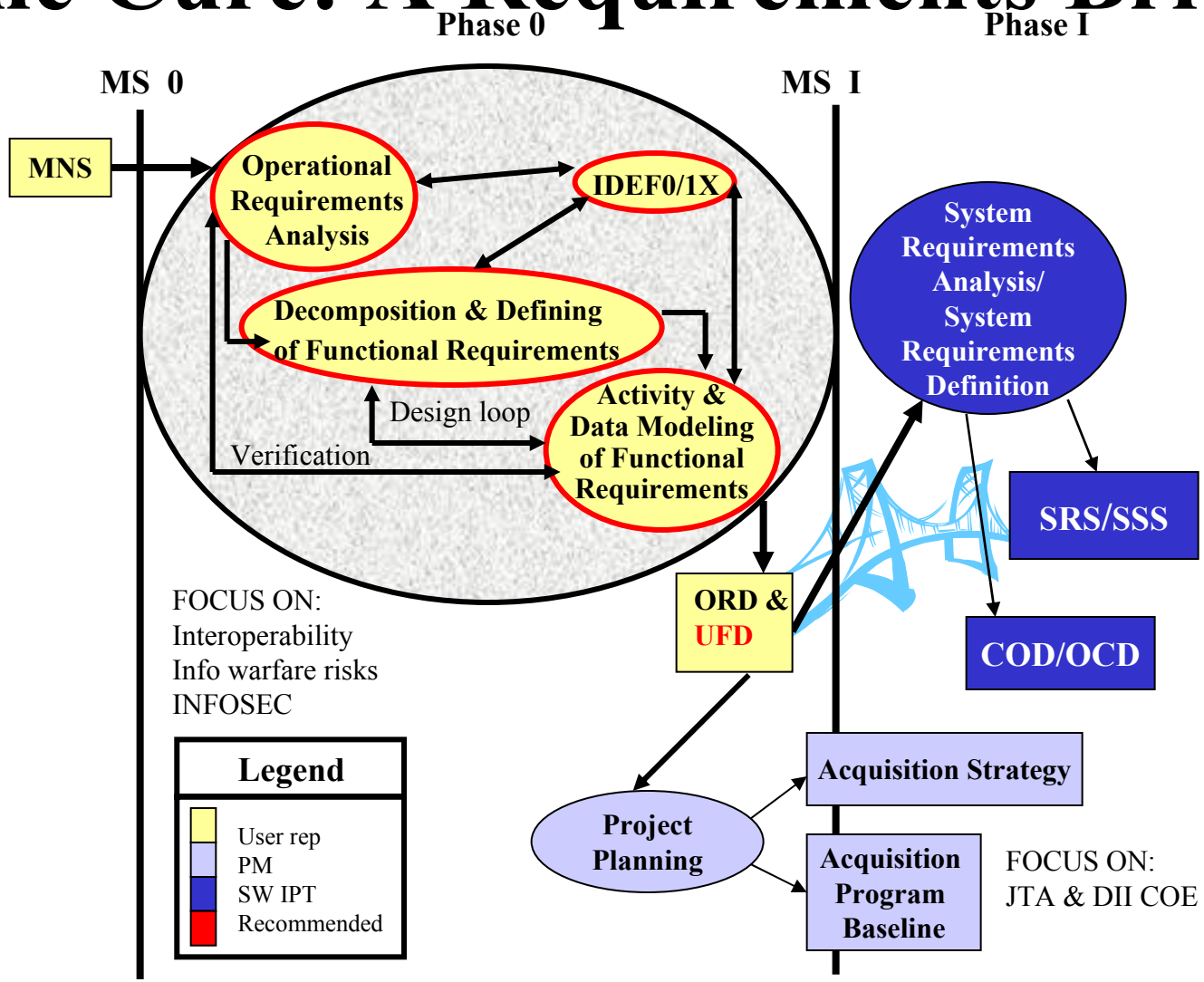
- A big cause of program failures.
- Inadequately stated requirements.
- Misunderstandings between users and developers and within the user community often contribute to the problem.
- When programs run into trouble, requirements (and/or) testing are often reduced.

Because

- Software is “soft” we often change it instead of bending metal... but...
- “it is not so soft that change is free....Change is the biggest money-maker in the software world.” Robert L. Glass.
- If requirements keep evolving as the software evolves it is next to impossible to develop a successful project. Developers find themselves shooting at a moving target and throwing away design and code faster than they can crank it out.


Classes of people, data sources to support
Query types, data loads, intelligence-terminal tradeoffs, response times
Internal data structure, buffer handling techniques
Detailed scheduling algorithms, error handling
Programmer understanding of specifications
EXAMPLE SOURCES OF UNCERTAINTY, MAN-MACHINE INTERFACE SOFTWARE
RELATIVE COST RANGE
4x
2x
1.5x
1.25x
x
0.8x
0.67x
.05x
0.25x
CONCEPT OF OPERATION
REQUIREMENTS SPECIFICATIONS
PRELIMINARY DESIGN SPECIFICATION
DETAILED DESIGN SPECIFICATION
ACCEPTED SOFTWARE
FEASIBILITY
PLANS AND REQUIREMENTS
PRODUCT DESIGN
PRODUCT DESIGN
DEVELOPMENT AND TEST
PHASE AND MILESTONES

The Cure: A Requirements Bridge



MNS: Mission Need Statement; **ORD:** Operational Requirements Document; **UFD:** Users' Functional Description; **JTA:** Joint Technical Architecture; **SRS:** System Requirements Specification; **SSS:** System/Subsystem Specification; **COD:** Concept of Operations Description; **OCD:** Operational Concept Description; **DII COE:** Defense Information Infrastructure Common Operating Environment

Users' Functional Description

Appendix O, TRADOC Pam 71-9

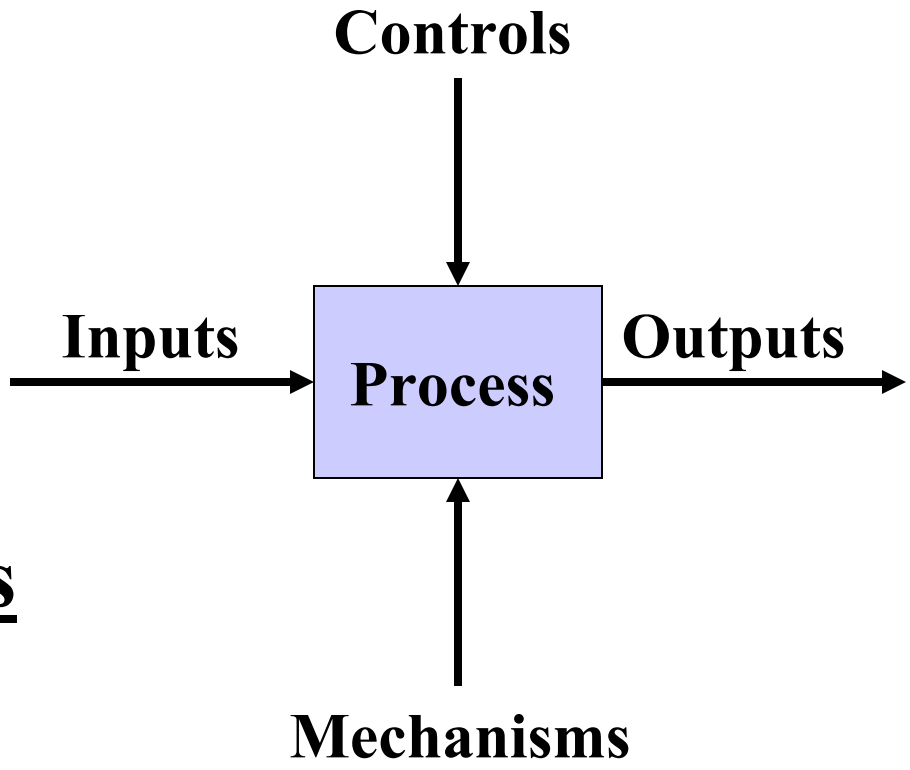
- The UFD is a follow-on document to the ORD, which specifically addresses requirements related to information technology (IT).
- The CBTDEV writes the UFD to refine the operational requirements for IT capabilities that were approved in the ORD.
- The UFD is approved by the proponent school commandant.
- The CBTDEV forwards the UFD to the MATDEV, Software Developer and Operational Tester.

Users' Functional Description

Appendix O, TRADOC Pam 71-9

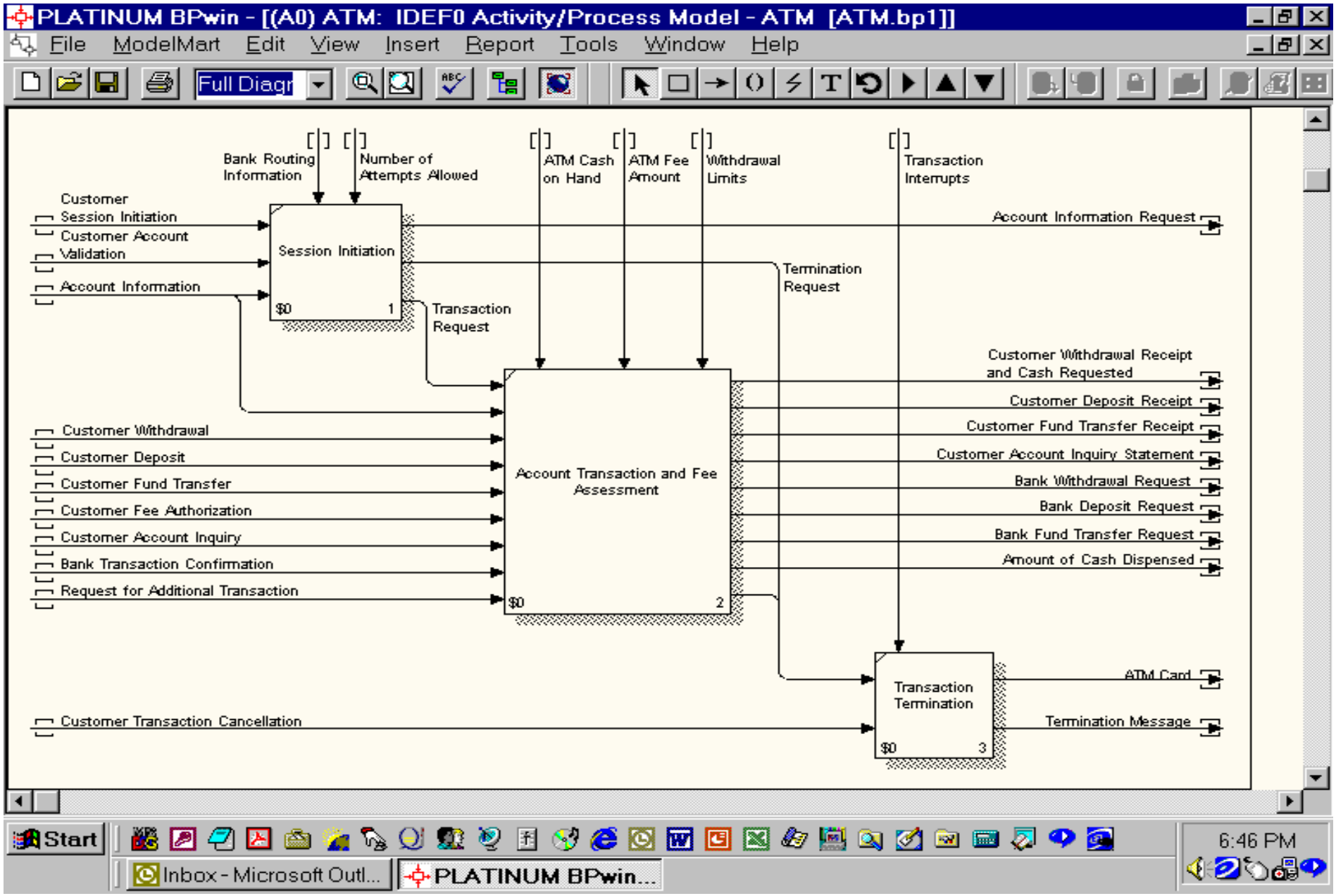
- **Section 3.0 Detailed Characteristics**
  - 3.1 Performance requirements
  - **3.2 *Functional requirements* - via IDEF0**
  - **3.3 *Inputs/outputs (data requirements)* - via IDEF1X**
  - 3.4 Failure contingencies
- **Section 4.0 Oper mode summary/mission profile**
- **Section 5.0 External Environments**
  - 5.1 Physical environment
  - 5.2 System architecture
  - 5.3 Organizational environment
  - 5.4 Threat environment

IDEF0 Process Model



ICOMs

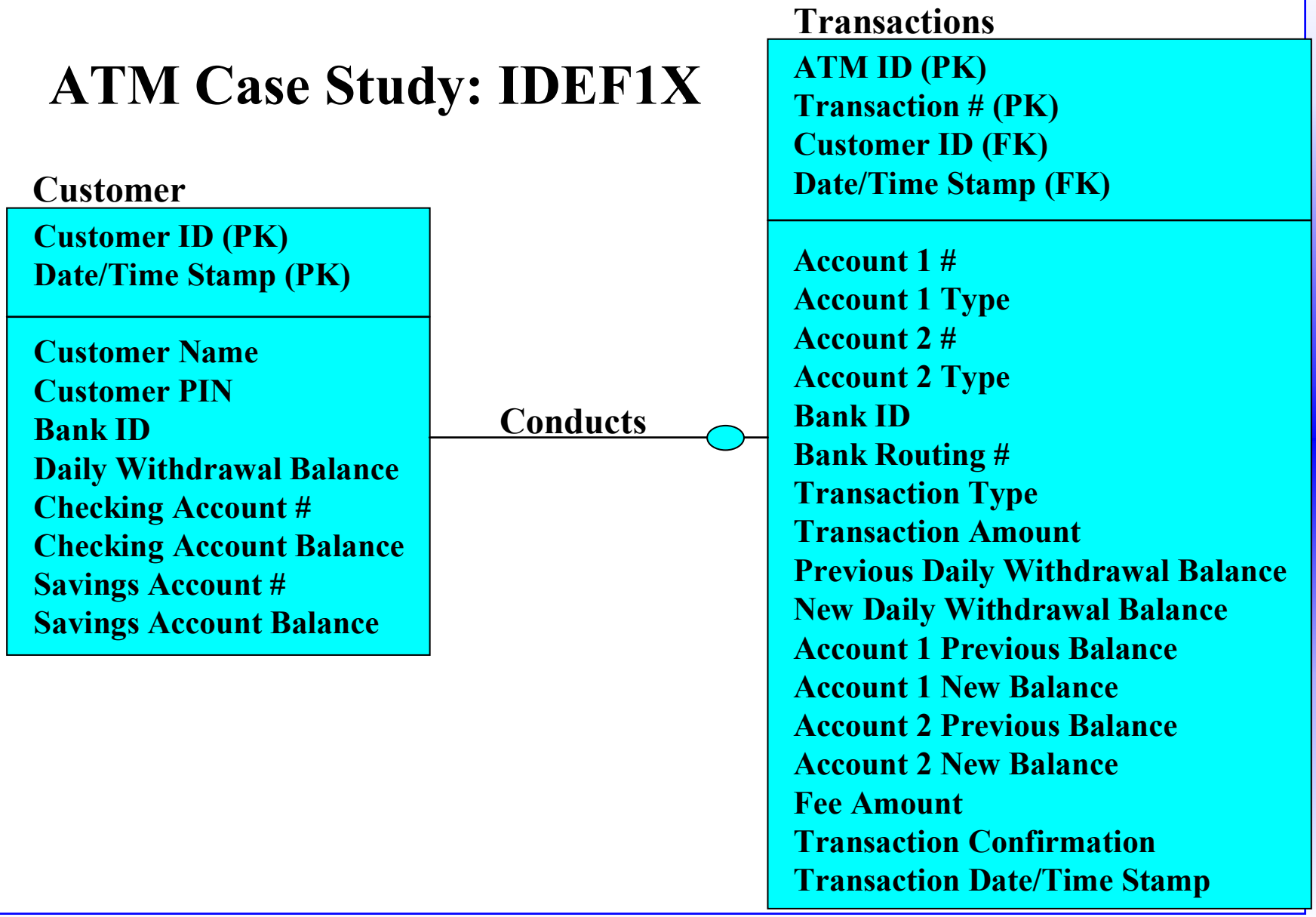
ATM Case Study: IDEF0



IDEF1X Data Modeling

1. **Create Conceptual Data Model**
   1) Identify entities (create a noun list).
   2) Determine relationship(s) between entities.
2. **Create Key Based Data Model**
   1) Identify primary key attribute(s).
   2) Identify foreign key attribute(s).
3. **Create Fully Attributed Data Model**
   1) Identify non-key attribute(s).

ATM Case Study: IDEF1X



More Software Blues

Dr. Delores Etter's (Deputy Under Secretary of Defense for Science and Technology) STC 1999 Keynote Address:

- **Half of all DoD software projects end up costing twice as much as originally estimated.**
- **DoD software projects suffer an average schedule slippage of three years.**

Cause 2

- **Inadequate Software Cost Estimates:**
  - **Incorrect software size estimates:**
    - ⇒ User representatives are incorrectly and incompletely defining requirements.
    - ⇒ Software size measures are critical.
      - A lack of training pervades DoD in the areas of software sizing and cost estimating.
  - **Requirements instability:**
    - Requirements creep.
    - Technology insertion.
    - Regulatory changes.

Software Size - Estimating Techniques

- Expert Judgment -
  - top-down technique, relies on experience/background/business sense
  - can overlook factors (overconfidence), lack of experience
- Delphi Judgment - iterative process
- Work Breakdown Structure - bottom-up
- Analogy
- Parametric
- Engineering

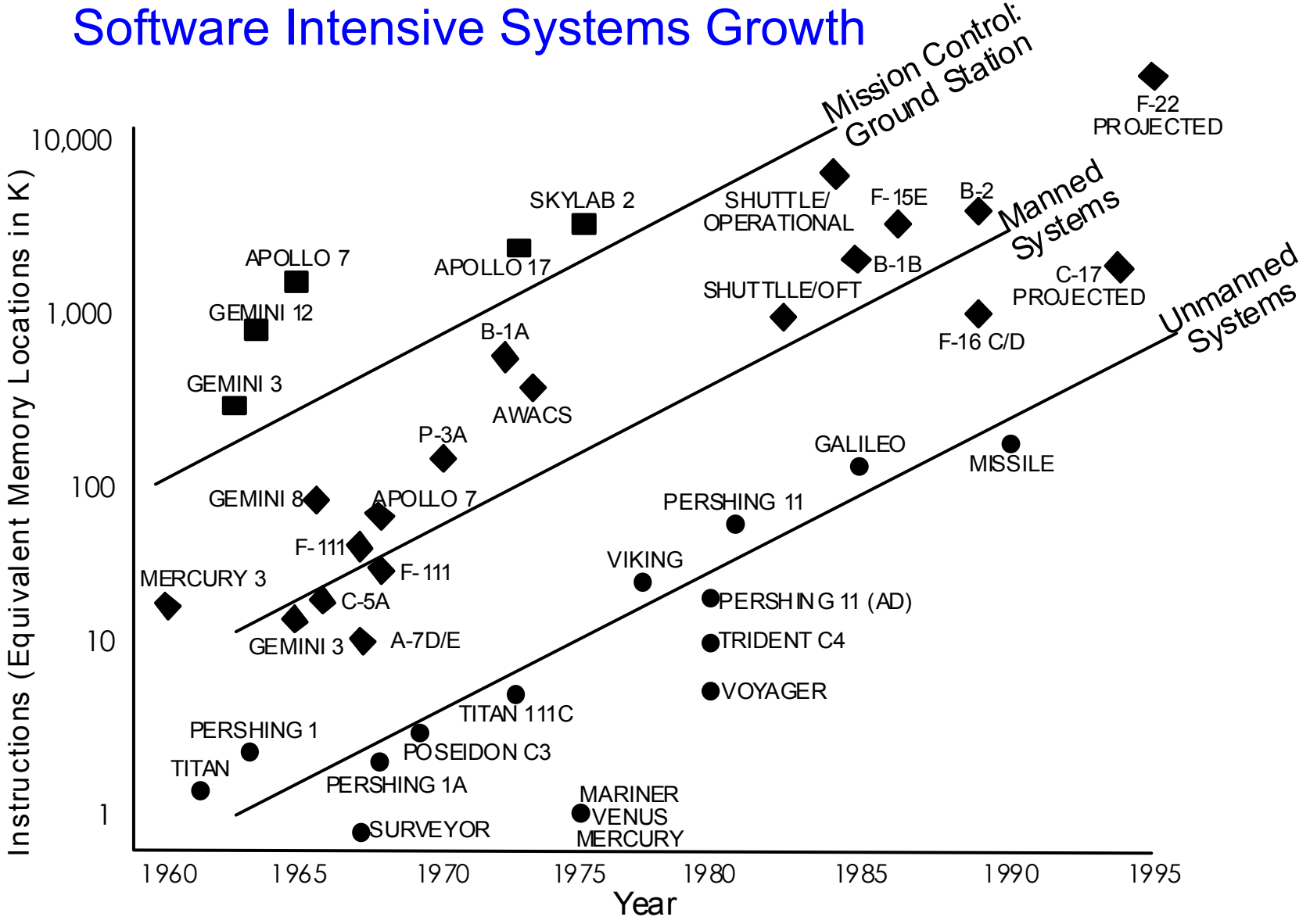
Software Sizing Options

MEASURE	ADVANTAGES	DISADVANTAGES
Function Points (FP)	Technology independent Logical user view International acceptance (IFPUG)	Requires training
Mark II Function Points	Technology independent Logical user view	Acceptance isolated-UK Proprietary (until recently) Dependent on implementation
Source Lines of Code (SLOC)	Can be done from physical implementation	Technology dependent Inconsistent rules Counter indicator of productivity Unavailable until coded
Feature Points (Capers Jones - early 1990's)	Enhanced FP for algorithmically intensive	Experimental
FP Backfired from LOC	Derive FP where limited documentation	Less accurate than FP Same limitations as LOC Applicable to Logical LOC only

Sizing Techniques

- Software cost estimating models today are based on
  - Lines of Code
  - Function Points
- Variations

Software Intensive Systems Growth



Details of one Parametric Software Cost Model

Development Effort (Man Months)

"Entropy" Constant, e.g..,

- Organic Mode: 1.05
- Semi-Detached Mode: 1.12
- Embedded Mode: 1.20

$$E = C \times S^a$$

"Size" of the software Product

Proportionality Constant…a product of cost, schedule, and personnel drivers like:

- Required Reliability
- Time Constraints
- Analyst Capability
- Programmer Experience
- Modern Programming Tools
- Software Tools Used
- etc

The problem lies not with the accuracy of the algorithms in the models, but with the inaccuracy of the size measurements fed into the models.

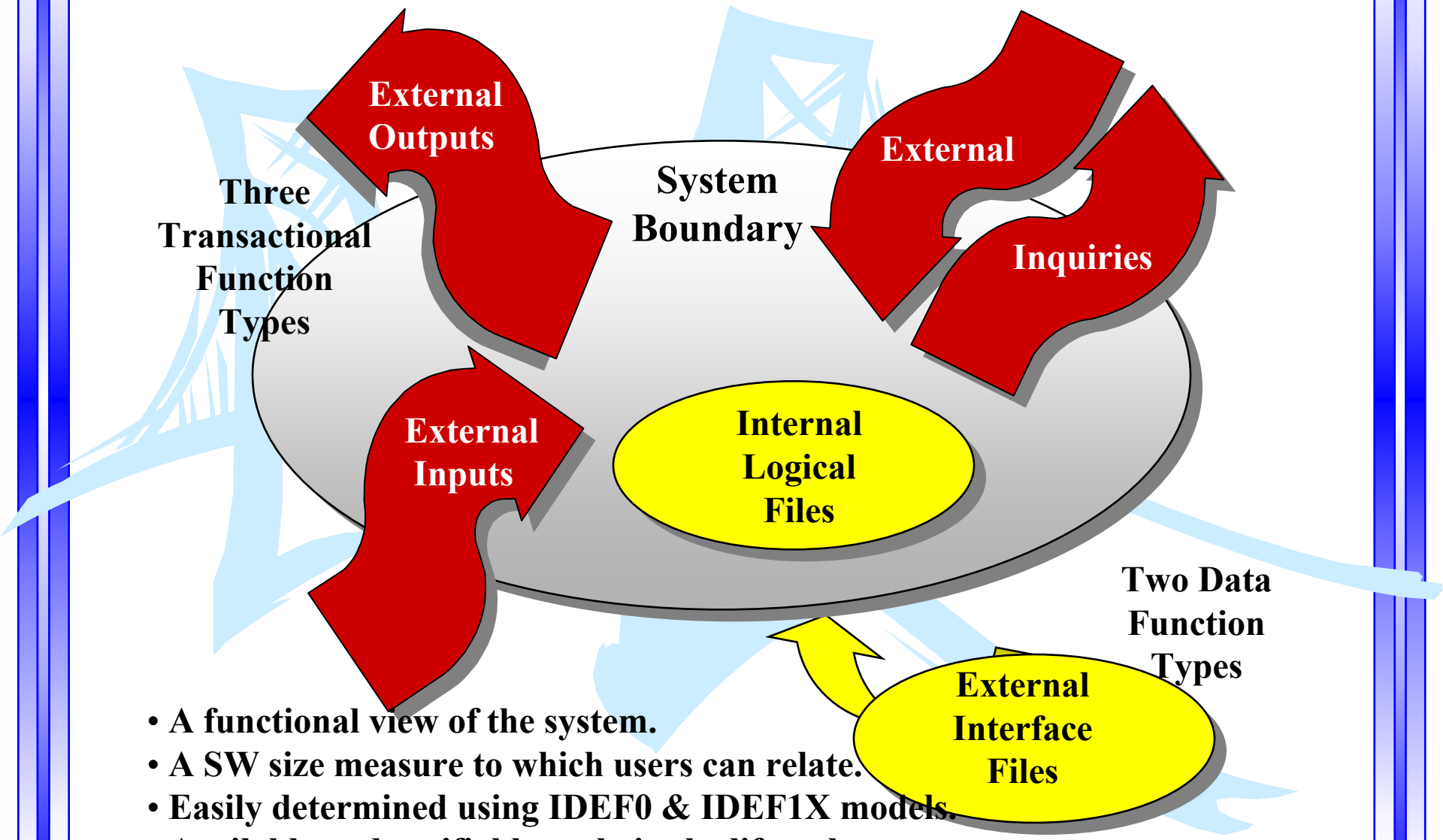
How to Use Parametric Models

- Size
- Environmental factors
  - Programmer Ability, Product Complexity, Product Size, Available Time, Required Reliability, Level of Technology, …
- Equations (Calibrated??)
- Output (Development and Maintenance)
  - In Terms of Effort, Schedule, Cost

Accuracy and Calibration

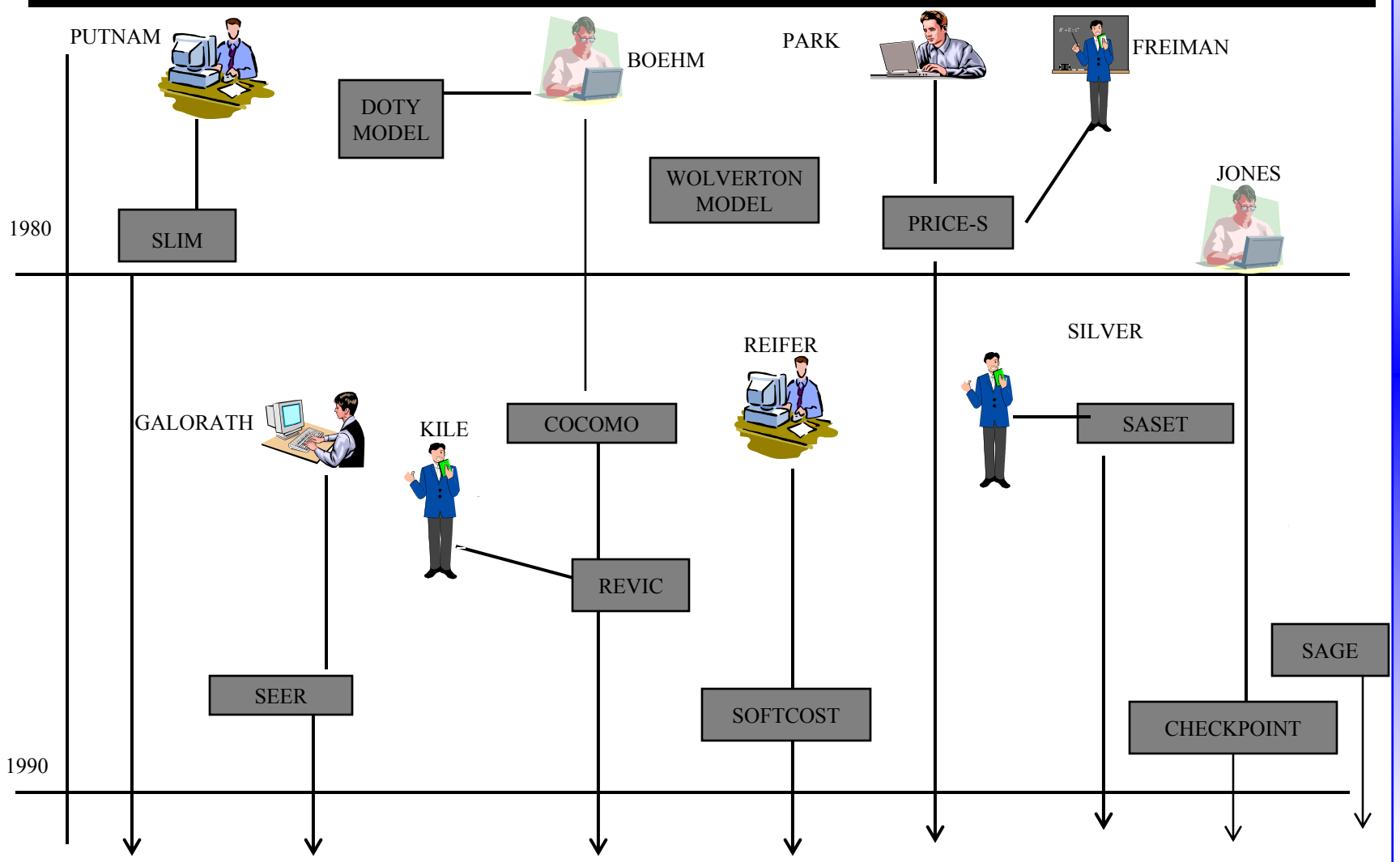
- Accuracy is directly proportional to the user's confidence in size estimates and the description of the environmental factors.
- Calibration is accomplished by refining the model parameters to reflect a particular type of development.
- REVIC's algorithms are developed using DOD systems.

Function Points



- A functional view of the system.
- A SW size measure to which users can relate.
- Easily determined using IDEF0 & IDEF1X models.
- Available and verifiable early in the lifecycle.

Software Cost Models



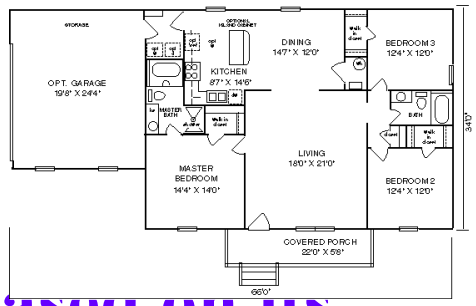
What are Function Points?

- **AA software *sizing measure* based on the functional (logical) user requirements to be delivered by software.**

Function Points can be:

- ***E*--*Estimated* at early requirements,**
  --*Counted* from known functional requirements, and
  --*Updated / refined* when the software is delivered or after functional changes during the software development life cycle.

What Are Function Points?



- FFPs measure the size of software **based on its Functional User Requirements (like Square Feet on a Floor Plan).**
- **DDevelopers build software based on requirements and perform hundreds of tasks before installation (like builders).**
- SSoftware Cost and Work Effort **is dependent on *size* *AND* *other attributes***

ADJUSTED FUNCTION POINTS

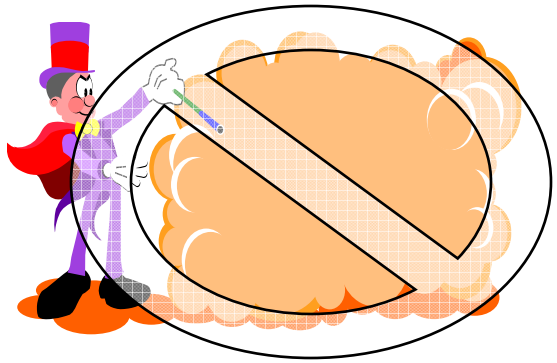
- **After the application's functions are counted: the unadjusted FP (UFP) is adjusted for other User /Business Constraints using a Value Adjustment Factor (VAF)**

- **The VAF is based on:**
  - ***Sum of 14 General System Characteristics (GSC) Questions, each rated 0 to 5***
  - ***Average application sum = approx. 35***

Facts about Function Points:

FPs *ARE NOT*

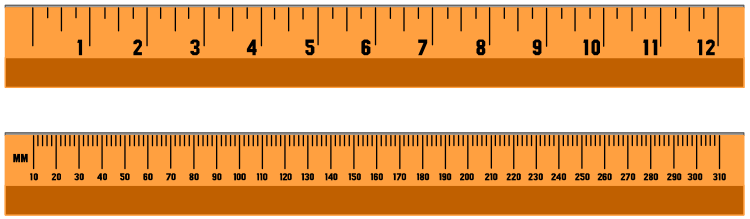
- **Sufficient by themselves to produce a cost or work effort estimate.**
- **A "silver bullet" measure.**
- **A quick fix or a solution to problems.**
- **A substitute for Project Attributes.**



Facts about Function Points:

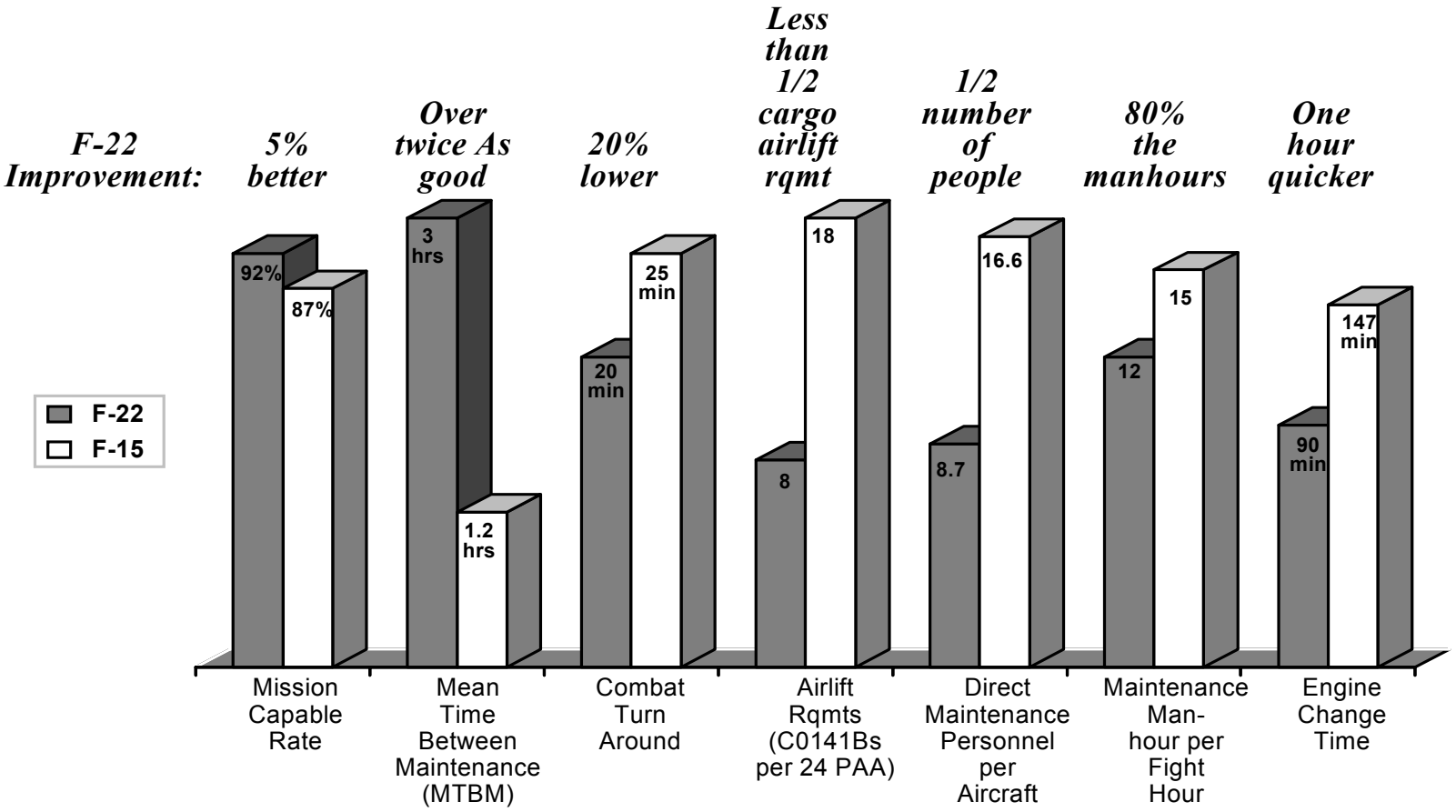
FPs *ARE*

- ☑ **A measure of *software size* based on its logical user requirements**
- ☑ **Can be used with additional project attributes to determine a software *cost estimate***
- ☑ ***Independent* of technology, tools, and other physical project attributes**



Success Stories? How about the F-22

F-22 versus F-15 Life Cycle Cost Comparison



Source: *Aviation Week & Space Technology*, July 24, 1995

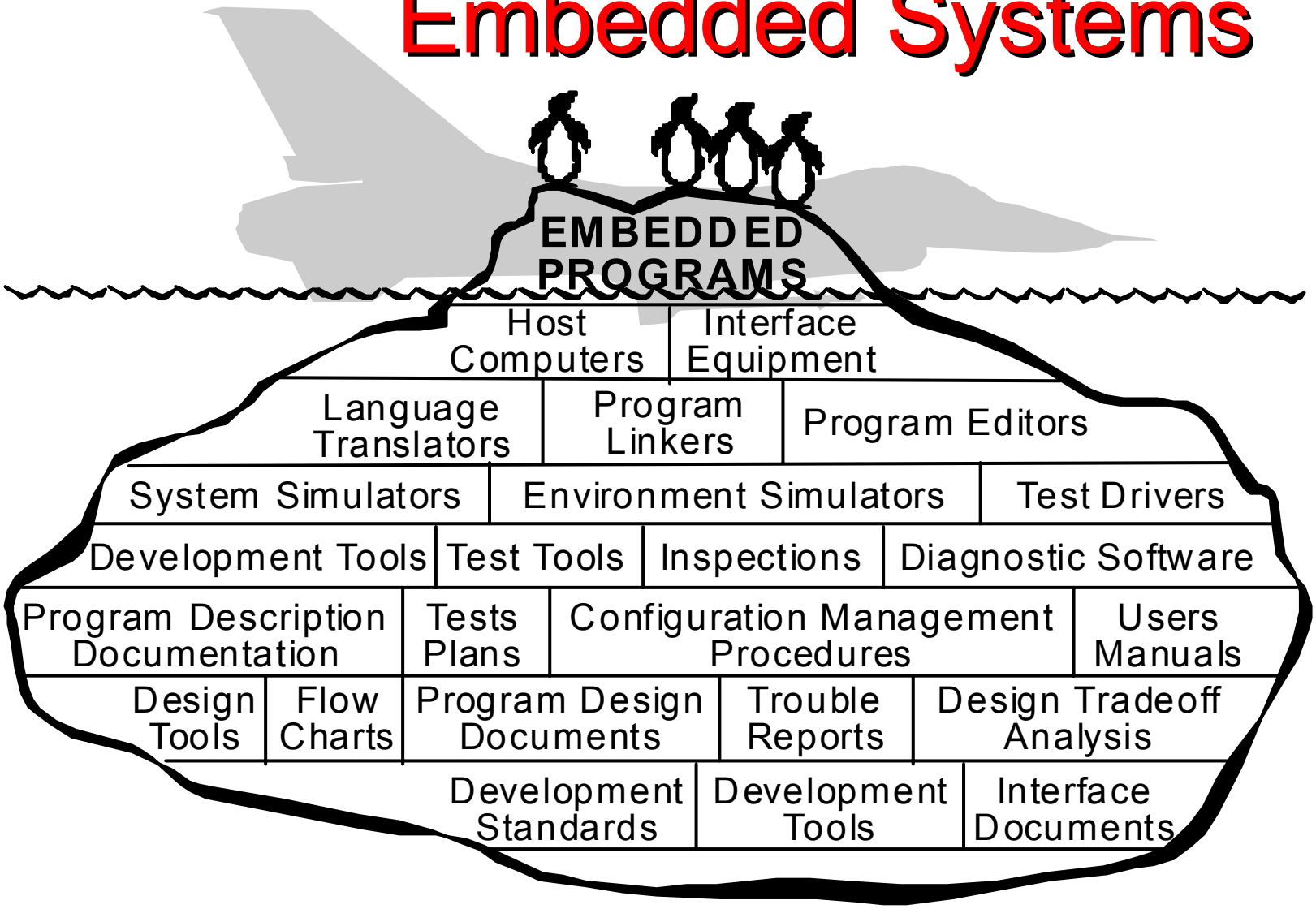
Boeing 777

- Seats 210-420 passengers, has complete fly by wire cockpit controls, has the most powerful engines ever built for an airliner, and was extensively tested and designed from scratch by software.
- $4 Billion, within cost and on schedule program using Ada.

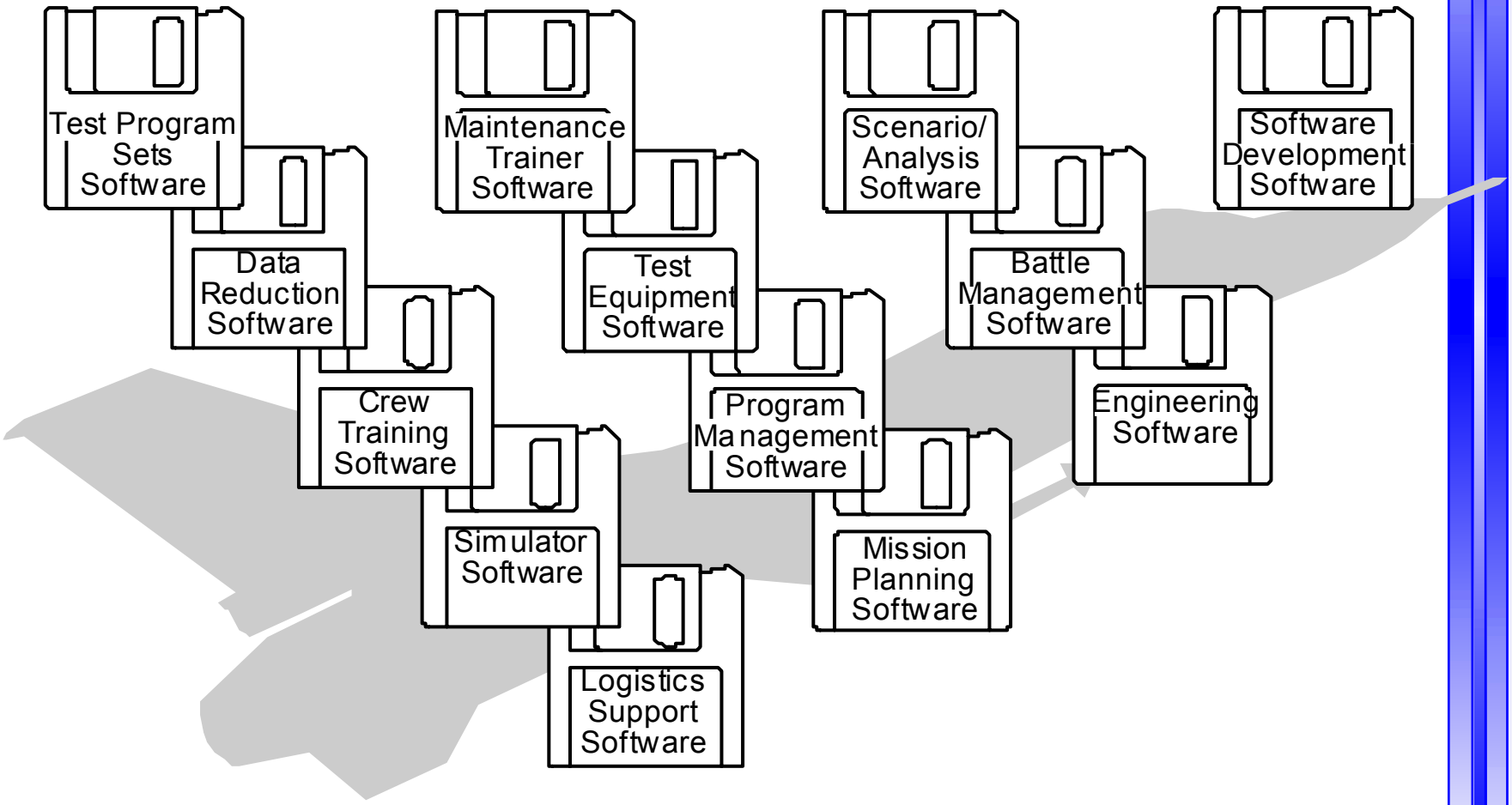
Optimistic Estimates

- Under pressure, contractors and PMs often make overly optimistic estimates about cost and schedule requirements.
- We often discard pessimistic cost, schedule, and size estimates and base our projections on the best of all possible worlds.
- We don't manage risk, build a management reserve, or a worst case scenerio into our cost schedules for fear our programs will not get funded or approved if we are more realistic.
- The problem is that we have realistic estimates... we just fail to use them for fear of the consequences... and then we run into trouble because we were not realistic.

Embedded Systems



Other Weapon System Software



C3I Software

Provides secure information to tactical operations

Software Development

- DOD 5000 basically lays out how we will develop systems including software.
- Guidance and requirements depend on the basic Acquisition Category that is assigned to the system.
- IEEE/EIA 12207 provides guidance.

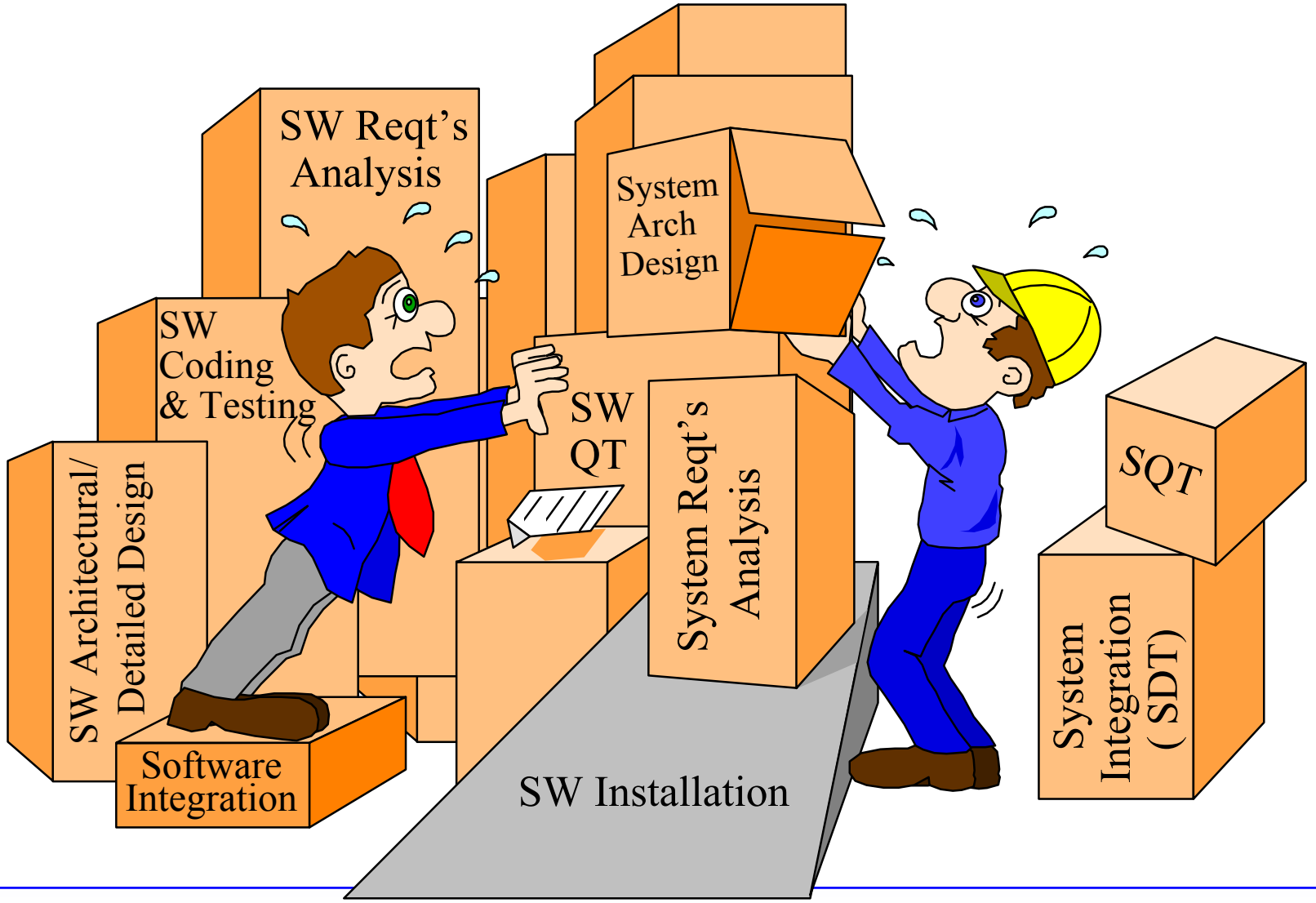
Software Life Cycle Processes IEEE/EIA 12207.0-1996

- What to do, not how to do it.
- Establishes a common framework for software life cycle processes.
- Meant to be tailored for each type of SW to which it is applied.
- Provides the "building blocks" needed to create a life-cycle model.

- Industry implementation of ISO/IEC 12207: 1995.

Software Development Process

"Building Blocks"



Languages

- Ada is here to stay. It is a proven language that has major safety and reliability advantages over conventional languages
- No policy requires the PM to use Ada.
- Examples of Ada successes.

Ada's Eagle : Our safest fighter

Air Force Success Story

YF-22

Picture This

Picture this. 12 subsystems, 650 modules, and millions of lines of code...stack ‘em up...And listen to this part: how long do you think it took to integrate this job? All this software. A year? Six months? No. Six weeks? Think again: 650 modules, 12 subsystems, and millions of lines-of code--a lot of code, and a lot of “stuff” and a lot of places- and a lot of states. ...How long do you think it took to integrate this software? If you guessed three days, you’re right. ..**T,H,R,E,E days**.

LTG Edmonds, USAF
YF-22 Prototype

Navy Success Story

Seawolf Submarine's
AN/BSY-2 Project

BSY-2

- Advanced, highly complex Ada system designed to handle all the signal and data processing of the Seawolf's inboard electronics suites.
- Takes real time data from thousands of sensors and converts, analyzes and sorts the data used by attack center
- 6+ million lines of code

Benefits of Reuse

- Increased productivity
- Shorter development time
- Reduced costs over time
- Increased quality and reliability
- Earlier requirements verification
- Lower risk through more accurate size estimates
- Higher ability to leverage expertise in individual domains
- Shorter time to field

Costs of Reuse

- The easiest way to reduce the number of SLOC is through reuse.
- Reuse costs:
  - Domain analysis and modeling
  - Domain architecture development
  - Inspection and quality assurance of reusable components
  - Increased documentation to facilitate reuse
  - Maintenance and enhancements of reusable assets
  - Training of personnel in design and coding for reuse

Software Configuration Management

- Keeping Control
- Process used to ID software configuration components and...
- System used for
  - Controlling change
  - Maintaining integrity

ADDITIONAL INFORMATION

- Martha Ann Spurlock
- (804) 765-4234
- Martha.Spurlock@dau.mil

THANK YOU!!!

QUESTIONS???